Message Strip-Mining Heuristics for High Speed Networks

Costin Iancu,

Parry Husbans,

Wei Chen

Motivation

- Increasing productivity: need compiler, run-time based optimizations. Optimizations need to be performance portable.
- Reducing communication overhead is an important optimization for parallel applications
- Applications written with bulk transfers or compiler may perform message "coalescing"
- Coalescing reduces message start-up time, but does not hide communication latency
- Can we do better?

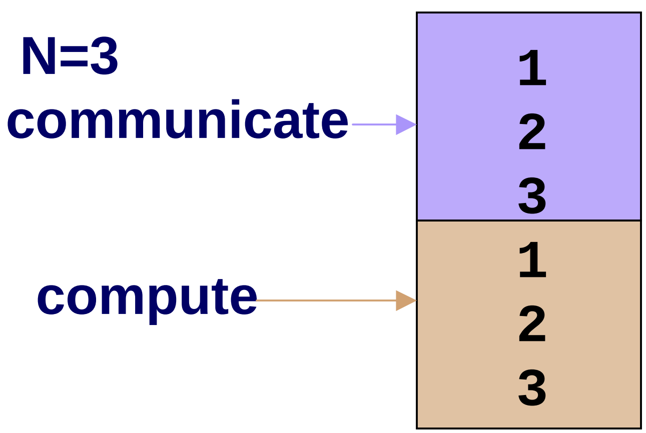
Message Strip-Mining

MSM (Wakatani) - divide communication and computation into phases and pipeline their execution

initial loop

N = # remote elts

```
shared [] double *p;
float *buf;
get(buf,p,N*8);
for(i=0;i<N;i++)
 …=buf[i];
```

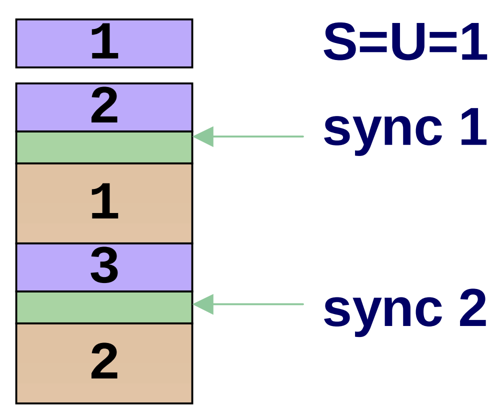


strip-mined loop

S = strip size

U = unroll depth

```
h0 = nbget(buf, p, S);
for(i=0; i < N; i+=S)
h1=nbget(buf+S*(i+1),p+S*(i+1),S);
    sync(h0);
    for(ii=i; ii < min(...); ii++)
          ...=buf[ii];
    h0=h1;
```



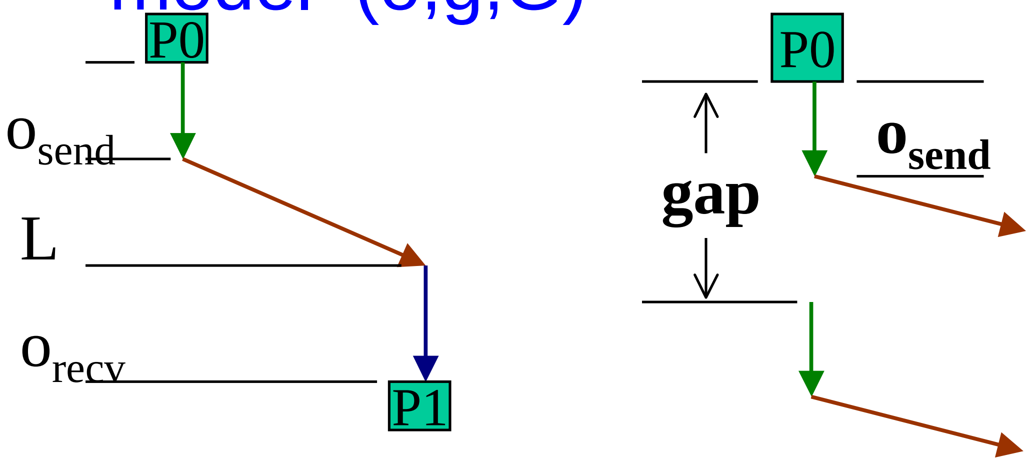
Performance Aspects of MSM

- Increased message start-up time, but potential for overlapping communication with computation. Unrolling increases message contention
- Goal: find heuristics that allow us to automate MSM in a performance portable way. Benefits both compiler based optimizations and “manual” optimizations

- Decomposition strategy dependent on:
  - system characteristics (network, processor, memory performance)
  - application characteristics (computation, communication pattern)
- How to combine?

Machine Characteristics

- Network performance: LogGP performance model (o,g,G)



MSM <-> o
unrolling <-> g

- Contention on the local NIC due to increased number of requests issued
- Contention on the local memory system due to remote communication requests (DMA interference)

Application Characteristics

- Transfer size - long enough to be able to tolerate increased start-up times (N,S)
- Computation - need enough available computation to hide the cost of communication (C(S))
- Communication pattern - determines contention in the network system (*one-to-one* or *many-to-one*)

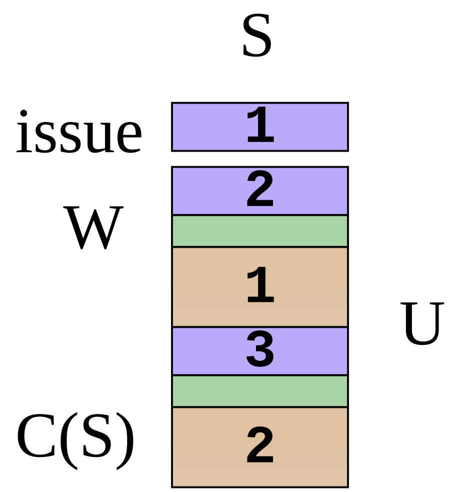
Questions

- What is the minimum transfer size that benefits from MSM?
- What is the minimum computation latency required?
- What is an optimal transfer decomposition?

Analytical Understanding

- Vectorized loop: $T_{vect} = o + G*N+C(N)$
- MSM + unrolling:

$$W(S_1) = G*S_1 - issue(S_2)$$
$$W(S_2) = G*S_2 - C(S_1) - W(S_1) - issue(S_3)$$
....
$$W(S_m) = G*S_m - C(S_{m-1}) - W(S_{m-1})$$



Minimize communication cost:

$$T_{strip+unroll} = \sum^{m} issue(S_i)+W(S_i)$$

Experimental Setup

System	Network	CPU
IBM Netfinity cluster	Myrinet 2000	866 MHZ Pentium PIII
IBM RS/6000	SP Switch 2	375 MHz Power 3+
Compaq Alphaserver ES45	Quadrics	1 GHz Alpha

- GasNet communication layer (performance close to native)
- Synthetic and application benchmarks
- Vary N - total problem size
  - S - strip size
  - U - unroll depth
  - P - number of processors
  - communication pattern

Minimum Message Size

- ***What is the minimum transfer size that benefits from MSM?***
  - Minimum cost is o+max(o,g)+ε
  - Need at least two transfers
  - Lower bound: N > max(o,g)/G
  - Experimental results : 1KB < N < 3KB
  - In practice: 2KB

Computation

- ***What is the minimum computation latency required to see a benefit?***
- Computation cost: cache miss penalties + computation time
- Memory Cost: compare cost of moving data over the network to the cost of moving data over the memory system.

System	Inverse Network Bandwidth (μsec/KB)	Inverse Memory Bandwidth (μsec/KB)	Ratio (Memory/Network)
Myrinet/PIII	6.089	4.06	67%
SPSwitch/PPC3+	3.35	1.85	55%
Quadrics/Alpha	4.117	0.46	11%

No minium exists: MSM always benefits due to memory costs

NAS Multi-Grid
(ghost region exchange)

Network	No Threads	Base (1)	Strip-Mining	Speed-up
Myrinet	2	1.24	0.81	1.53
	4	0.71	0.49	1.45
SP Switch	2	0.69	0.42	1.64
	4	0.44	0.35	1.25
Quadrics	2	0.32	0.28	1.14
	4	0.29	0.28	1.03

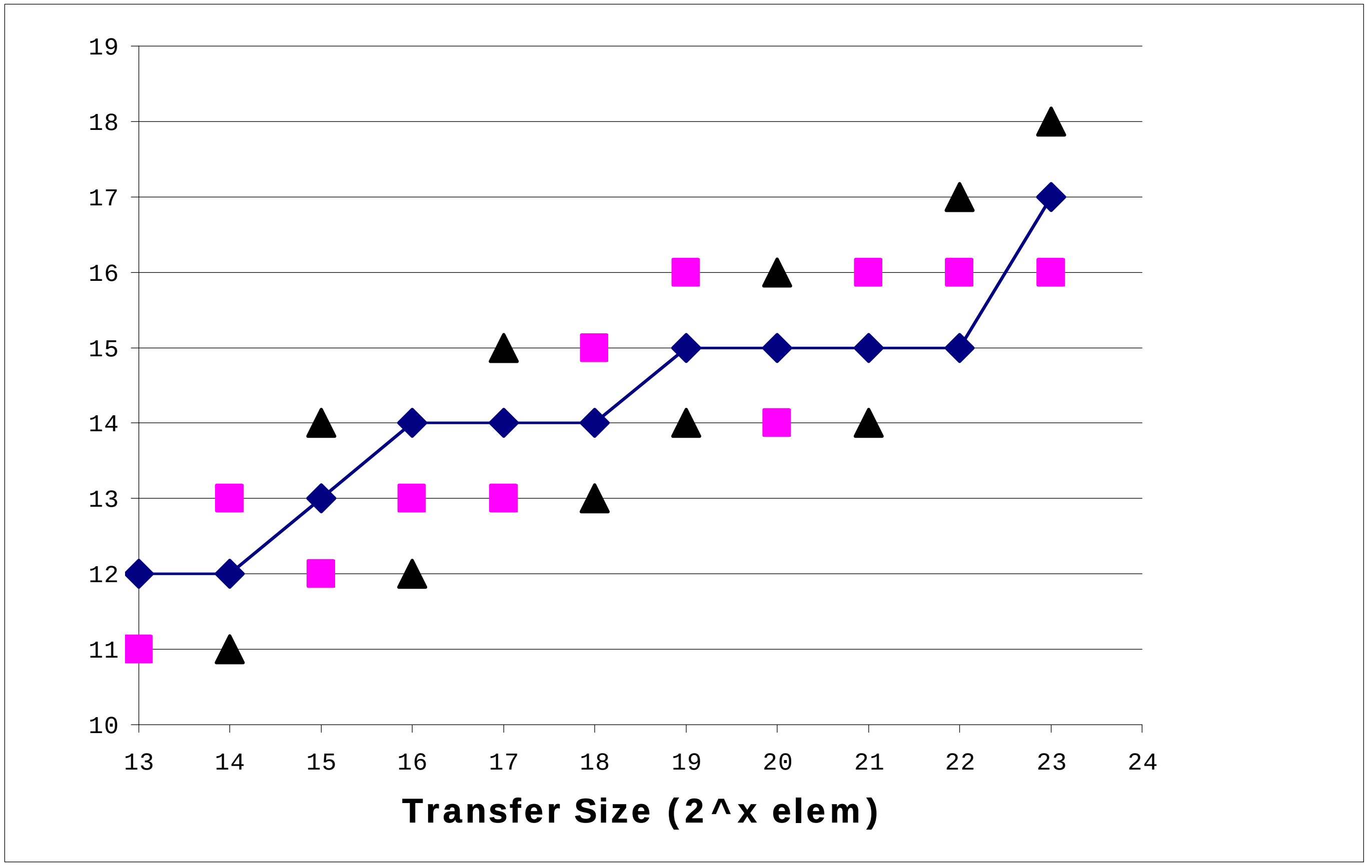
Decomposition Strategy

- ***What is an optimal transfer decomposition?***
  - transfer size - ***N***
  - computation - $C(S_i) = K*S_i$
  - communication pattern - ***one-to-one, many-to-one***
- Fixed decomposition: simple. Need to search the space of possible decompositions.
- Not optimal overlap due to oscillations of waiting times.
- Idea: try a variable block-size decomposition
- Block size continuously increases $S_i = (1+f)*S_{i-1}$
- How to determine values for ***f*** ?

Benchmarks

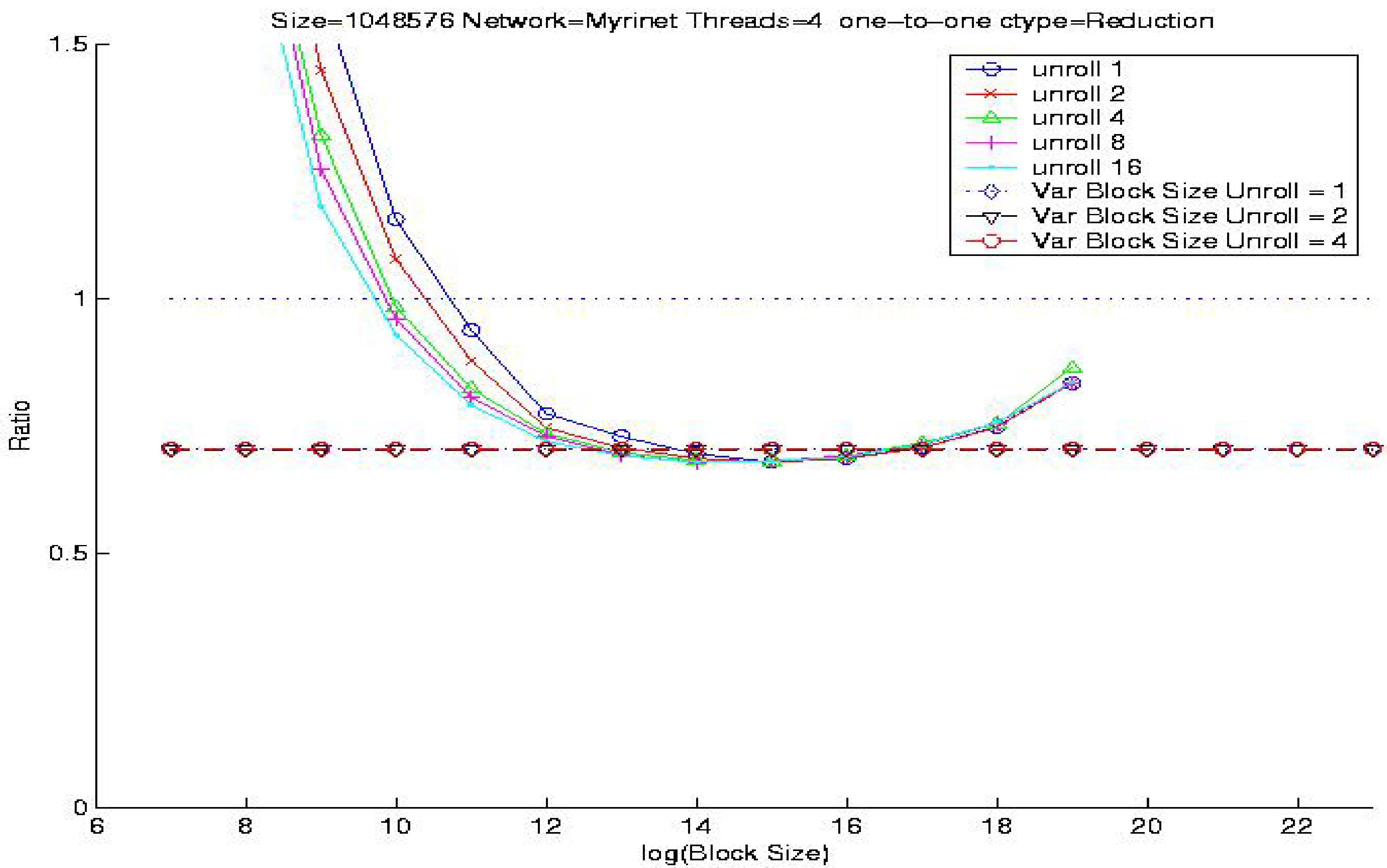
- Two benchmarks
  - Multiply accumulate reduction (same order of magnitude with communication) (***C(S) = G*S***)
  - Increased computation (~20X) (***C(S) = 20*G*S***)
- Total problem size ***N:*** 2^8 to 2^{20} (2KB to 8MB)
- Variable strip decomposition ***f*** tuned for the Myrinet platform. Same value used over all systems

Transfer Size

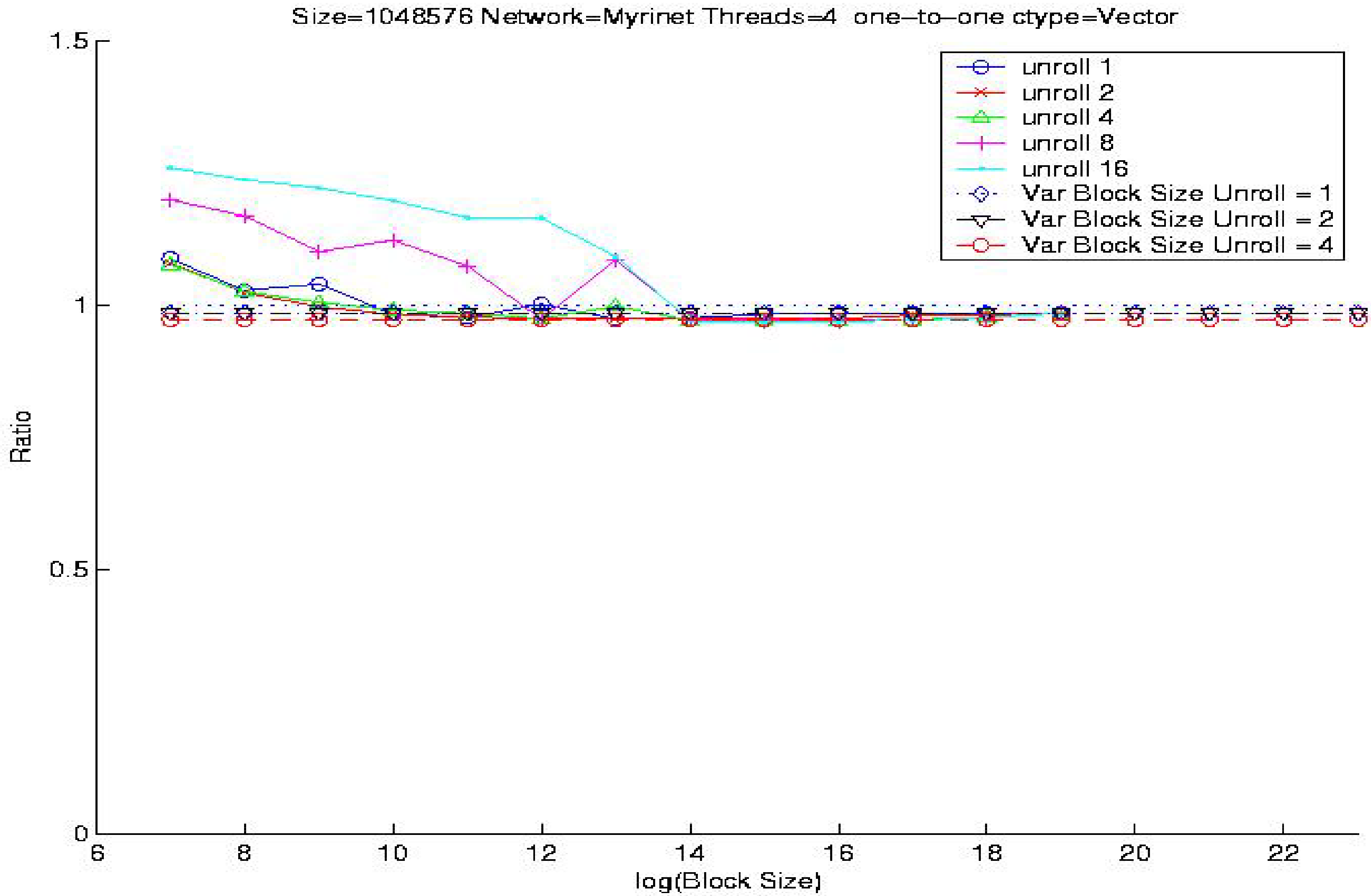


Variation of size for optimal decomposition (Myrinet)
MAC reduction

Computation:
MAC Reduction



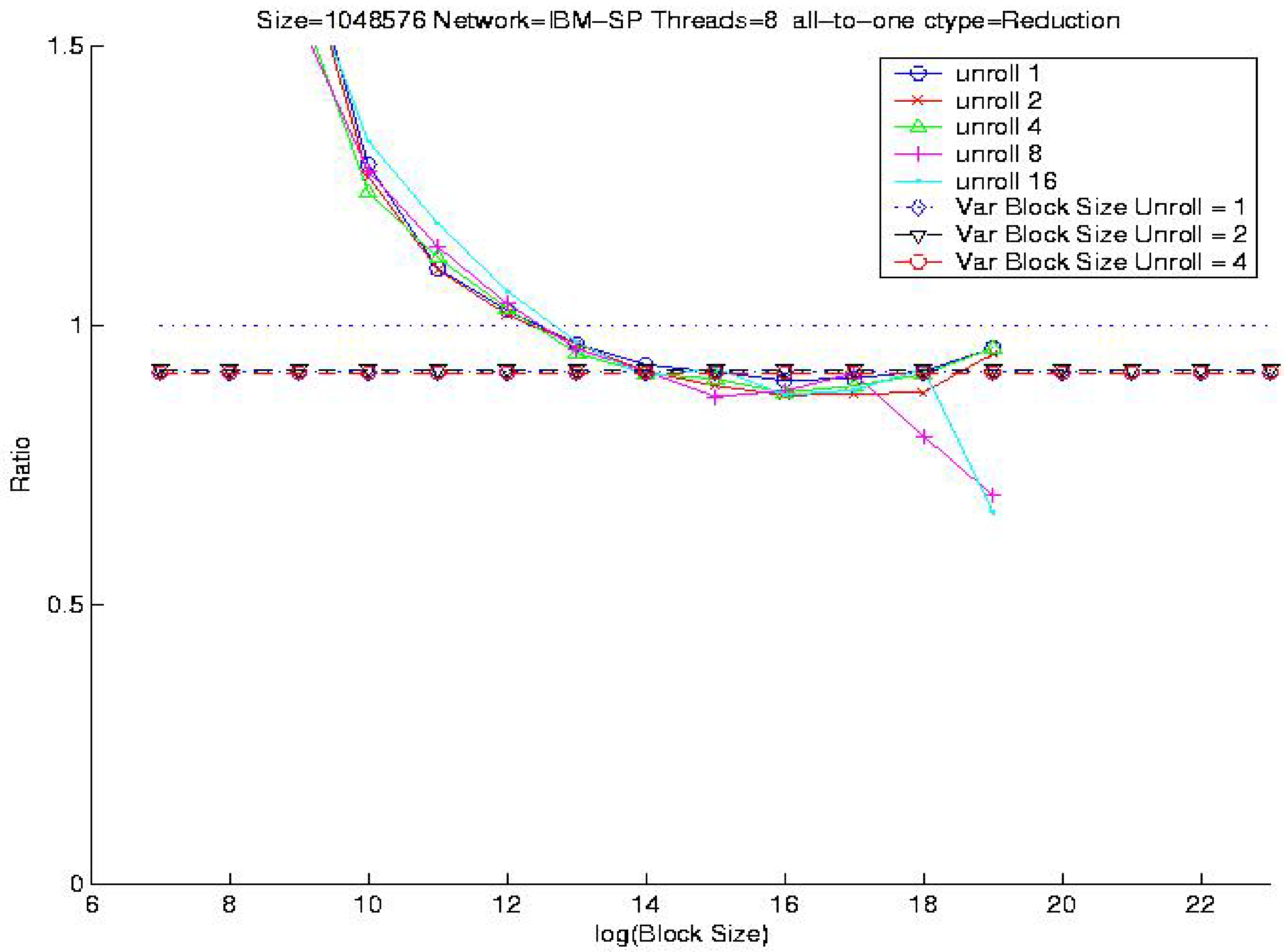
Increased Computation



Communication Pattern

- Contention on the memory system and NIC
- Memory system: measure slowdown of computation on “node” serving communication requests
- 3%-6% slowdown
- NIC contention - resource usage and message serialization

Network Contention



Summary of Results

- MSM improves performance, able to hide most communication overhead

- Variable size decomposition is performance portable (0%-4% on Myrinet, 10%-15% with un-tuned implementations)
- Unrolling influenced by g. Not worth with large degree (U=2,4)
- For more details see full paper at http://upc.lbl.gov/publications

MSM in Practice

- Fixed decomposition - performance depends on **N/S**
- Search decomposition space. Prune based on heuristics: N↑-S↑, C↑-S↓, P↑-S↑
- Requires retuning for any parameter change
- Variable size - performance depends on ***f***
- Choose ***f*** based on memory overhead (0.5) and search. Small number of experiments

Implications and Future Work

- Message decomposition for latency hiding worth applying on a regular basis
- Ideally done transparently through run-time support instead of source transformations.
- Current work explored using only communication primitives on contiguous data. Same principles apply for strided/"vector" accesses - need unified performance model for complicated communication operations
- Need to combine with a framework for estimating the optimality of compound loop optimizations in the presence of communication - benefits all PGAS languages

END

Performance Aspects of MSM

- MSM - decompose large transfer into stripes, transfer of each stripe overlapped with communication
- Unrolling increases overlap potential by increasing the number of messages that can be issued
- However:
  - MSM increases message startup time
  - unrolling increases message contention
- How to combine? - determined by both **hardware** and **application** characteristics